

FACULTY OF ENGINEERING AND TECHNOLOGY BACHELOR OF TECHNOLOGY

High Performance Computing (HPC) (203105430)

VII SEMESTER

Computer Science & Engineering Department



Logo

Description automatically generated

CERTIFICATE

*This is to certify that*

*Mr..* **LUCKY PATHAN** *with Enrollment No.* **210303105790** has *successfully completed her laboratory experiments in the* **High Performance Computing (203105430)** *from the department of* **Computer Science and Engineering** *during the academic year* ***2023-2024.***



**Date of Submission …..…………. Staff In charge …..……………**

**Head of Department ……………..**

**TABLE OF CONTENT:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **SR NO.** | **PRACTICAL LIST** | **START DATE** | **END DATE** | **SIGN** | **MARKS** |
| 1 | Study the facilities provided by Google Colab. |  |  |  |  |
| 2 | Demonstrate basic Linux Commands |  |  |  |  |
| 3 | Using Divide and Conquer Strategies design a class for Concurrent Quick Sort using C++. |  |  |  |  |
| 4 | Write a program on an unloaded cluster for several different numbers of nodes and record the time taken in each case. Draw a graph of execution time against the number of nodes. |  |  |  |  |
| 5 | Write a program to check task distribution using Gprof. |  |  |  |  |
| 6 | Use Intel V-Tune Performance Analyzer for Profiling. |  |  |  |  |
| 7 | Analyze the code using Nvidia-Profilers. |  |  |  |  |
| 8 | Write a program to perform load distribution on GPU using CUDA. |  |  |  |  |
| 9 | Write a simple CUDA program to print ͞Hello World!͟ |  |  |  |  |
| 10 | Write a CUDA program to add two arrays. |  |  |  |  |

# PRACTICAL- 1

AIM: **Study the Facilities provided by Google Colab.**

Google Colab is a free cloud-based platform for data analysis, research, and machine learning. With Colab, researchers can access powerful hardware, develop and run Python code in a Jupyter notebook environment, and share and collaborate on projects with ease.

Researchers should be aware of Colab's limits, which include a monthly utilization cap on CPU, GPU, and TPU usage as well as a session runtime cap and memory restrictions. Colab is nonetheless a potent tool for academics despite these drawbacks, offering a user-friendly design, compatibility for widely used libraries, and simple access to resources.

**What is a Notebook in Google Colab?**

In Google Colab, a notebook is a web-based environment for creating and running code. Notebooks are similar to scripts or code files in other programming environments but offer some unique advantages. Notebooks allow you to write and execute code in a web browser, displaying the output in real time. This makes it easy to iterate on your code and visualize the results as you go. Colab notebooks also support markdown, allowing you to include formatted text, equations, and images alongside your code. You can also add comments and notes to your code, which makes it easier to understand and collaborate with others. Overall, notebooks are a powerful tool for data scientists and machine learning practitioners, providing a flexible and interactive environment for writing and testing code.

**Google Colab Features:-**

* Colab provides users free access to GPUs and TPUs, which can significantly speed up the training and inference of machine learning and deep learning models.
* Colab’s interface is web-based, so installing any software on your local machine is unnecessary. The interface is also intuitive and user-friendly, making it easy to get started with coding.
* Colab allows multiple users to work on the same notebook simultaneously, making collaborating with team members easy. Colab also integrates with other Google services, such as Google Drive and GitHub, making it easy to share your work.
* Colab notebooks support markdown, which allows you to include formatted text, equations, and images alongside your code. This makes it easier to document your work and communicate your ideas.
* Colab comes pre-installed with many popular libraries and tools for machine learning and deep learning, such as TensorFlow and PyTorch. This saves time and eliminates the need to manually install and configure these tools.

**Advantages :-**

Users can build and run Python programs on the free cloud-based Google Colaboratory platform, also known as Colab. Machine learning, data analysis, and research are its main uses. Using Colab has a number of major advantages, including:

* Powerful hardware is available to customers through Colab, which can considerably speed up the training of machine learning models.
* Teams can easily work together on a project using Colab notebooks, which allow for easy collaboration and sharing.
* Google Drive integration allows Colab notebooks to be loaded from and saved to Google Drive, making it simple to access and share files.
* Free to use: There are no setup fees or usage charges when using Colab.
* Popular libraries are supported by Colab, making it simple to start working on machine learning
* projects. These libraries include TensorFlow, PyTorch, and Keras.
* Convenient interface: The data science and machine learning community frequently uses the Jupyter notebook interface, which Colab offers.

**Alternatives:-**

Google Colaboratory (Colab) can be used for machine learning, data analysis, and research, although there are a number of alternatives. Here are a few well-liked choices:

* **Kaggle Kernels:** Users of the cloud-based Kaggle Kernels platform can create and run code in a Jupyter notebook setting. It has a sizable user base and is largely used for data science and machine learning competitions.
* **2 Microsoft Azure Notebooks:** Users can create and use Jupyter notebooks on the cloud-based platform known as Microsoft Azure Notebooks. Both Azure Machine Learning services and access to powerful hardware are incorporated.
* **Jupyter Notebook:** Jupyter Notebook is an open-source web tool that enables users to create and share documents with live code, equations, visuals, and text. You have the option of running it locally on your computer or online.
* **Databricks:** A cloud-based platform that enables the creation and execution of Apache Spark workloads. It offers a collaborative setting for analytics, machine learning, and data engineering.
* **IBM Watson Studio:** A cloud-based tool called IBM Watson Studio enables users to create, develop, and employ machine learning models. It integrates with IBM's existing AI services and offers access to potent hardware like GPUs and TPUs.
* **6 DataCamp:** DataCamp is a cloud-based platform that offers practice problems for data science and machine learning as well as interactive coding training.

**GPUs and TPUs on Google Colab**

Ask anyone who uses Colab why they love it. The answer is unanimous – the availability of free GPUs and TPUs. Training models, especially deep learning ones, takes numerous hours on a CPU. We’ve all faced this issue on our local machines. GPUs and TPUs, on the other hand, can train these models in a matter of minutes or seconds.

*If you still need a reason to work with GPUs, check out*[*this excellent explanation*](https://www.analyticsvidhya.com/blog/2017/05/gpus-necessary-for-deep-learning/?utm_source=blog&utm_medium=google-colab-machine-learning-deep-learning)*by Faizan Shaikh.*

It gives you a decent GPU for free, which you can continuously run for 12 hours. For most data science folks, this is sufficient to meet their computation needs. Especially if you are a beginner, then I would highly recommend you start using Google Colab.

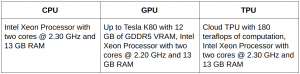
Google Colab gives us three types of runtime for our notebooks:

* CPUs,
* GPUs, and
* TPUs

As I mentioned, Colab gives us 12 hours of continuous execution time. After that, the whole virtual machine is cleared and we have to start again. We can run multiple CPU, GPU, and TPU instances simultaneously, but our resources are shared between these instances.

Let’s take a look at the specifications of different runtimes offered by Google

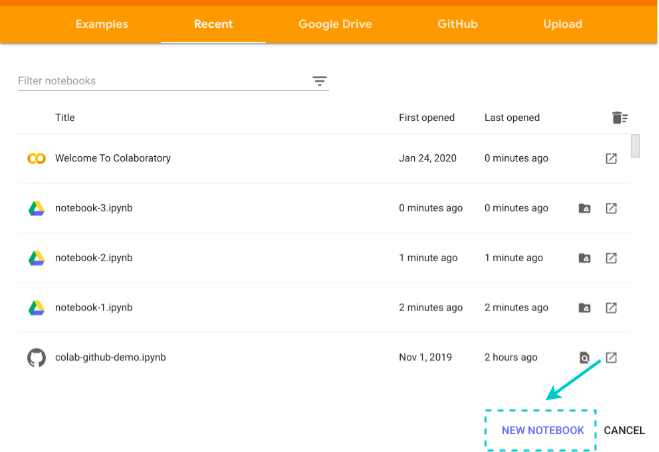
Colab:

[](https://cdn.analyticsvidhya.com/wp-content/uploads/2020/03/uc2-e1583900164853.png)

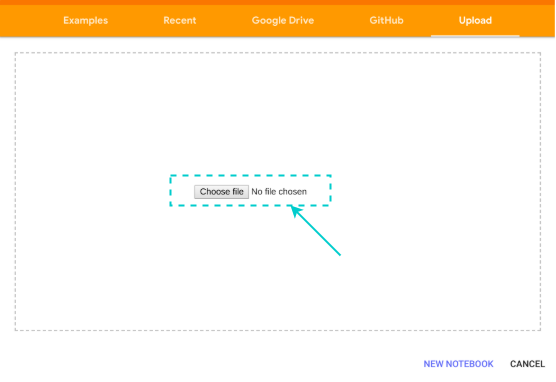
It will cost you A LOT to buy a GPU or TPU from the market. Why not save that money and use Google Colab from the comfort of your own machine?

**How to Use Google Colab?**

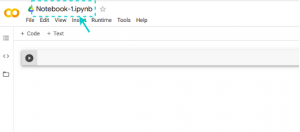
You can go to Google Colab using [this link](https://colab.research.google.com/). This is the screen you’ll get when you open Colab:

[](https://cdn.analyticsvidhya.com/wp-content/uploads/2020/03/uc1.png)

Click on the **NEW NOTEBOOK** button to create a new Colab notebook. Upload your local notebook to Colab by clicking the upload button:

[](https://cdn.analyticsvidhya.com/wp-content/uploads/2020/03/uc17.png)

You can also import your notebook from Google Drive or GitHub, but they require an authentication process.

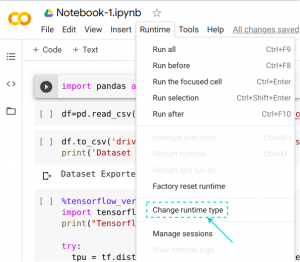
[](https://cdn.analyticsvidhya.com/wp-content/uploads/2020/03/uc3.png)

You can rename your notebook by clicking on the notebook name and change it to anything you want. I usually name them according to the project I’m working on.

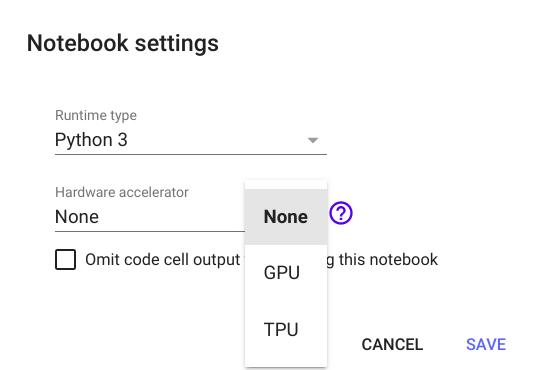
**Google Colab Runtimes – Choosing the GPU or TPU Option**

The ability to choose different types of runtimes is what makes Colab so popular and powerful. Here are the steps to change the runtime of your notebook:

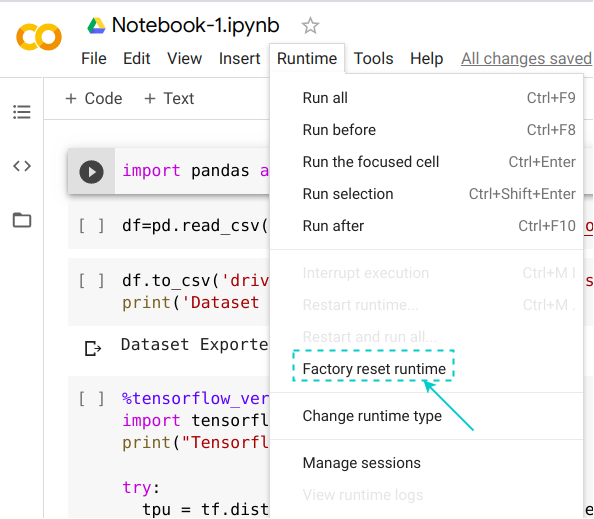
**Step 1:** Click ‘Runtime’ on the top menu and select ‘Change Runtime Type’:

[](https://cdn.analyticsvidhya.com/wp-content/uploads/2020/03/uc14.png)

**Step 2:** Here you can change the runtime according to your need:

[](https://cdn.analyticsvidhya.com/wp-content/uploads/2020/03/uc4.png)

A wise man once said, “With great power comes great responsibility.” I implore you to shut down your notebook after you have completed your work so that others can use these resources because various users share them. You can terminate your notebook like this:

[](https://cdn.analyticsvidhya.com/wp-content/uploads/2020/03/uc15-e1583920445689.png)

Using Terminal Commands on Google Colab

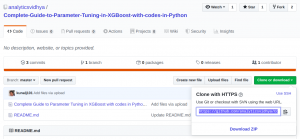
You can use the Colab cell for running terminal commands. Most of the popular libraries come installed by default on Google Colab. Yes, [Python](https://courses.analyticsvidhya.com/courses/introduction-to-data-science?utm_source=blog&utm_medium=google-colab-machine-learning-deep-learning) libraries like [Pandas](https://courses.analyticsvidhya.com/courses/pandas-for-data-analysis-in-python?utm_source=blog&utm_medium=google-colab-machine-learning-deep-learning), NumPy, [scikit-learn](https://courses.analyticsvidhya.com/courses/get-started-with-scikit-learn-sklearn?utm_source=blog&utm_medium=google-colab-machine-learning-deep-learning) are all pre-installed.

If you want to run a different [Python library](https://www.analyticsvidhya.com/blog/2019/07/dont-miss-out-24-amazing-python-libraries-data-science/?utm_source=blog&utm_medium=google-colab-machine-learning-deep-learning), you can always install it inside your Colab notebook like this: !pip install **library\_name**

Pretty easy, right? Everything is similar to how it works in a regular terminal. We just you have to put an **exclamation(!)** before writing each command like: !ls or: !pwd

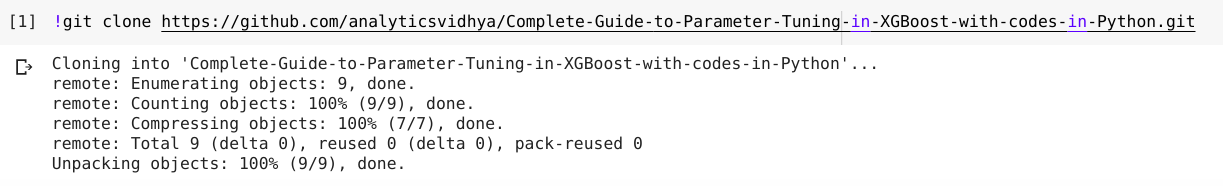
**Cloning Repositories in Google Colab:-**

You can also clone a Git repo inside Google Colaboratory. Just go to your [GitHub repository](https://www.analyticsvidhya.com/blog/category/github/?utm_source=blog&utm_medium=google-colab-machine-learning-deep-learning) and copy the clone link of the repository:

[](https://cdn.analyticsvidhya.com/wp-content/uploads/2020/03/uc5.png)

Then, simply run:

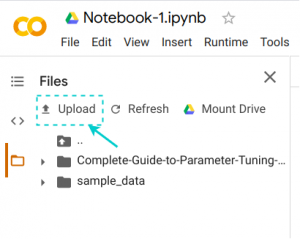
!git clone https://github.com/analyticsvidhya/Complete-Guide-to-Parameter-Tuning-in-XGBoost-with-codes-in-Python.git

[](https://cdn.analyticsvidhya.com/wp-content/uploads/2020/03/uc6.png)

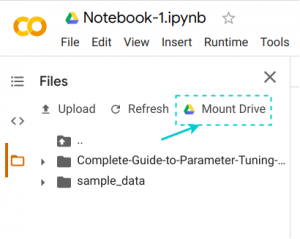
**Uploading Files and Datasets:-**

Here’s a must-know aspect for any data scientist. The ability to import your dataset into Colab is the first step in your data analysis journey.

The most basic approach is to upload your dataset to Colab directly:

[](https://cdn.analyticsvidhya.com/wp-content/uploads/2020/03/uc7.png)

You can use this approach if your dataset or file is very small because the upload speed in this method is quite low. Another approach that I recommend is to upload your dataset to Google Drive and mount your drive on Colab. You can do this in just one click of your mouse:

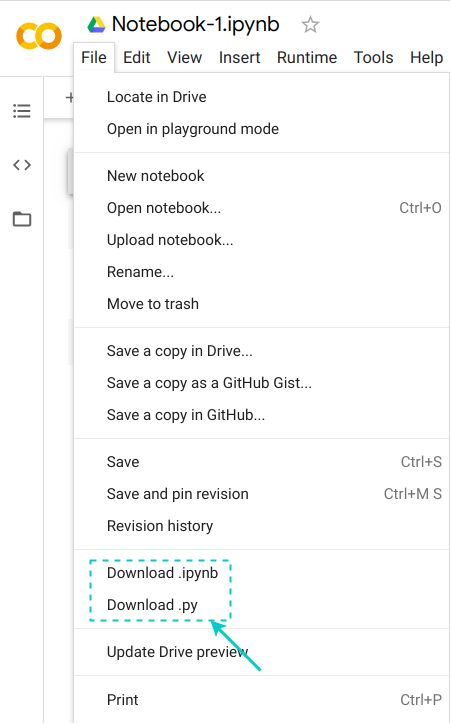
[](https://cdn.analyticsvidhya.com/wp-content/uploads/2020/03/uc8.png)

You can also upload your dataset to any other platform and access it using its link. I tend to go with the second approach more often than not (when feasible).

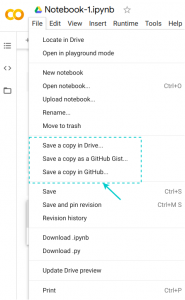
**Saving Your Notebook:-**

All the notebooks on Colab are stored on your Google Drive. The best thing about Colab is that your notebook is automatically saved after a certain time period and you don’t lose your progress.

If you want, you can export and save your notebook in both \*.py and \*.ipynb formats:

[](https://cdn.analyticsvidhya.com/wp-content/uploads/2020/03/uc13.png)

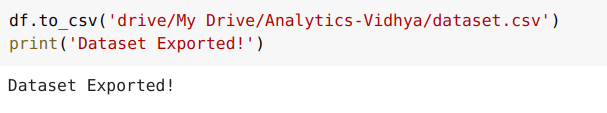
Not just that, you can also save a copy of your notebook directly on GitHub, or you can create a GitHub Gist:

[](https://cdn.analyticsvidhya.com/wp-content/uploads/2020/03/uc10.png)

I love the variety of options we get.

**Exporting Data/Files from Google Colab:-**

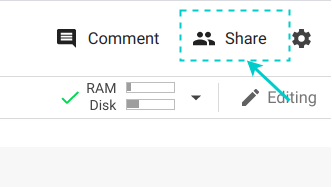
You can export your files directly to Google Drive, or you can export it to the VM instance and download it by yourself:

[](https://cdn.analyticsvidhya.com/wp-content/uploads/2020/03/uc9.png)

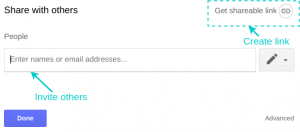
Exporting directly to the Drive is a better option when you have bigger files or more than one file. You’ll pick up these nuances as you work on bigger projects in Colab.

**Sharing Your Notebook:=**

Google Colab also gives us an easy way of sharing our work with others. This is one of the best things about Colab:

[](https://cdn.analyticsvidhya.com/wp-content/uploads/2020/03/uc11-e1583901050742.png)

Just click the **Share** button, and it gives us the option of creating a shareable link that we can share through any platform. You can also invite others using their email IDs. It’s exactly the same as sharing a Google Doc or Google Sheet. The intricacies and simplicity of Google’s ecosystem are astounding!

[](https://cdn.analyticsvidhya.com/wp-content/uploads/2020/03/uc12.png)

# PRACTICAL- 2

AIM: **Demonstrate basic Linux Commands.**

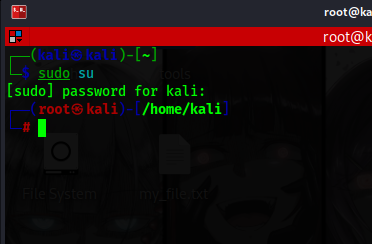
**Features :**

1. Customizability: Linux has a wide range of customization options, enabling users to modify their operating system to suit their own requirements and preferences. Users have the option to customize their Linux experience, from desktop settings to programme options.
2. Stability and Reliability: Linux is renowned for its stability and dependability, which makes it a top choice for servers and critical systems. There are few system breakdowns or malfunctions thanks to its sturdy architecture and effective memory management, which allow seamless operation even under heavy workloads.
3. Security: Strong security features are a hallmark of Linux. Due to its open-source design, flaws can be easily found and fixed by the community. Linux systems are less vulnerable to malware and cyberattacks because they contain strong built-in security procedures, such as user permissions, secure shell (SSH), and firewall setups.
4. Open-Source: Linux adheres to the principles of open-source software by letting users access, alter, and share the source code. This promotes a thriving community of programmers and fans who work together to continuously enhance the operating system. Transparency and user empowerment are also supported by the open-source philosophy.
5. Wide Variety of programmer: Linux supports a huge selection of software programmers, from programming environments to server programmers, from productivity tools to multimedia software. Whatever your computer needs, whether you're a researcher, a developer, or a casual user, Linux has a wide range of apps to meet them.

**Commands :**

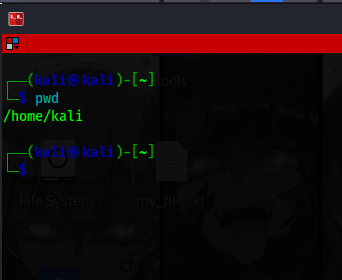
1. **Sudo:**

Short for superuser do, sudo lets you perform tasks that require administrative or root permissions.

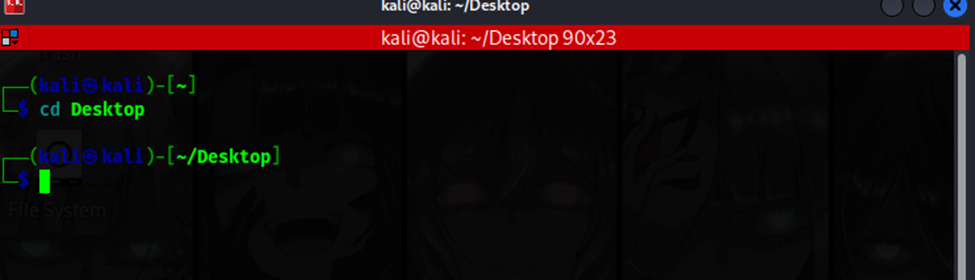


1. **Pwd:**

Use the pwd command to find the path of your current working directory.



1. **Cd:**

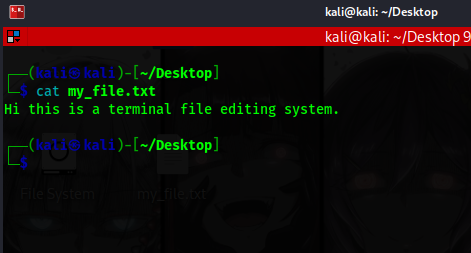
To navigate through the Linux files and directories, use the cd command.

1. **Ls:**

The ls command lists files and directories within a system.

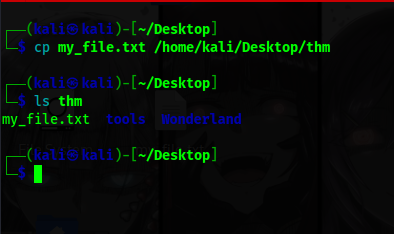
1. **Cat:**

It lists, combines, and writes file content to the standard output.



1. **Cp:**

Use the cp command to copy files or directories and their content.



1. **Mv:**

The primary use of the mv command is to move and rename files and directories.



1. **Mkdir:**

Use the mkdir command to create one or multiple directories at once and set permissions for each of them.

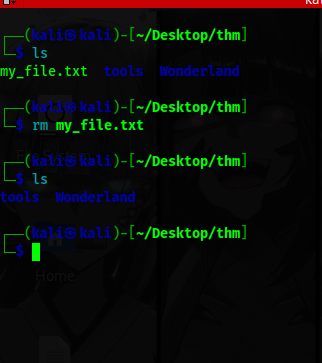
1. **Rmdir:**

To permanently delete an empty directory, use the rmdir command.



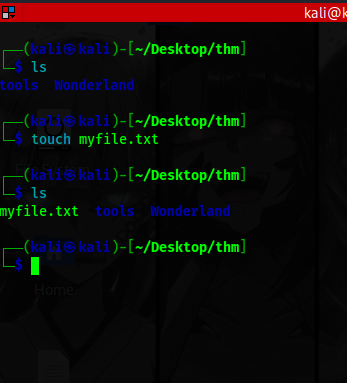
1. **Rm:**

The rm command is used to delete files within a directory.



1. **Touch:**

The touch command allows you to create an empty file or generate and modify a timestamp in the Linux command line.

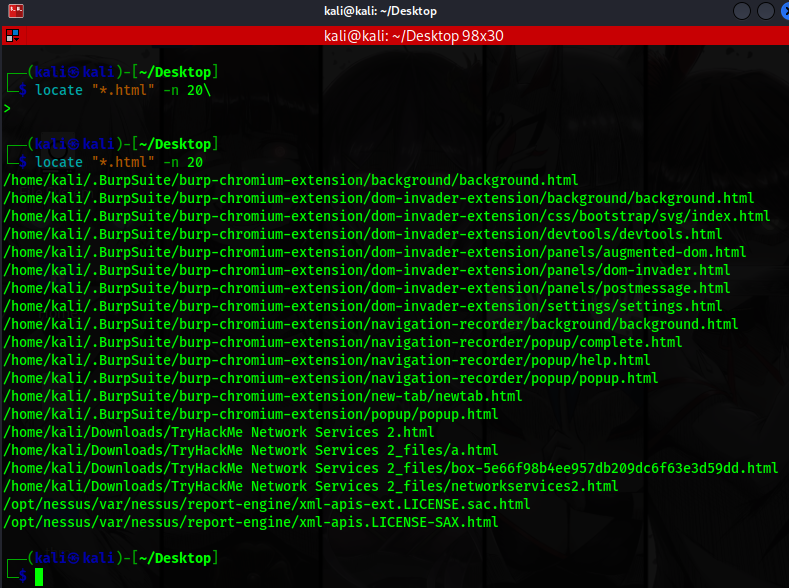


1. **Find:**

Use the find command to search for files within a specific directory and perform subsequent operations.

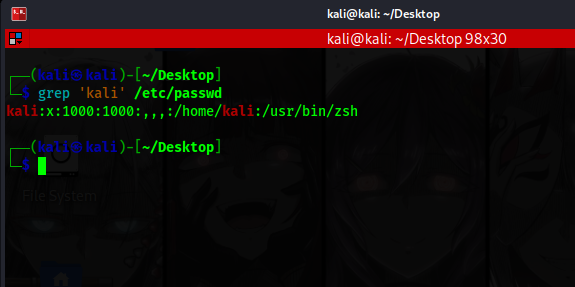


1. **Locate:**

The locate command can find a file in the database system.

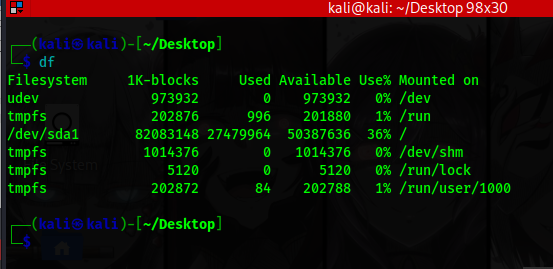
1. **Grep:**

Another basic Linux command on the list is grep or global regular expression print.

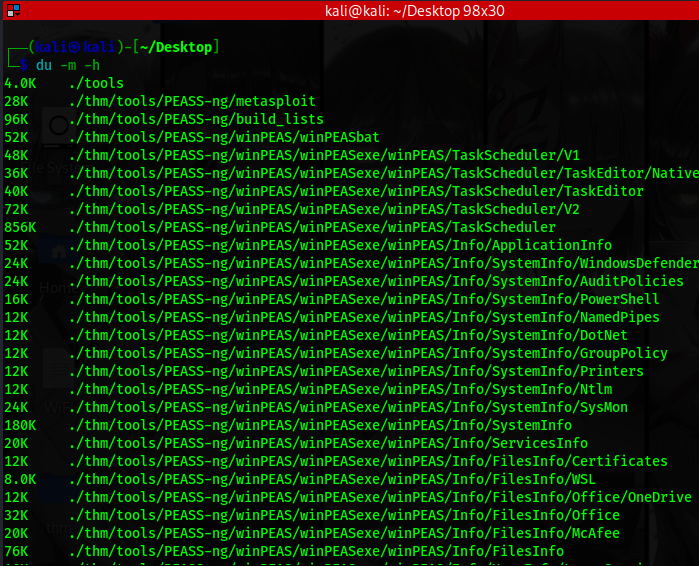


1. **Df:**

Use the df command to report the system’s disk space usage, shown in percentage and kilobyte (KB).

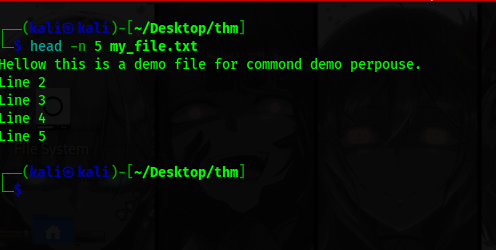


1. **Du:**

If you want to check how much space a file or a directory takes up, use the du command.

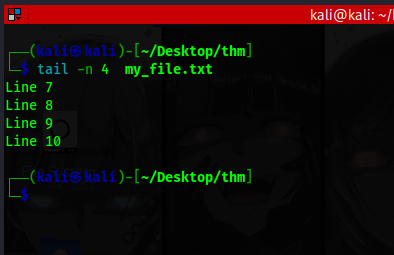
1. **Head:**

The head command allows you to view the first ten lines of a text.



1. **Tail:**

The tail command displays the last ten lines of a file. It allows users to check whether a file has new data or to read error messages.



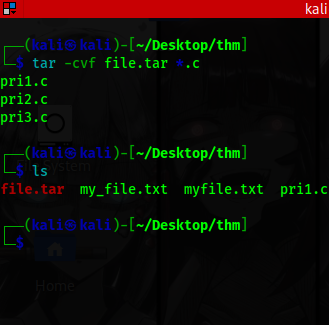
1. **Diff:**

The diff command compares two contents of a file line by line.



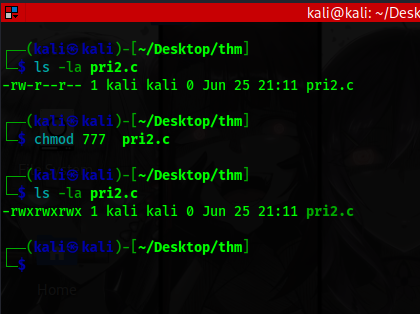
1. **Tar:**

The tar command archives multiple files into a TAR file – a common Linux format similar to ZIP, with optional compression.



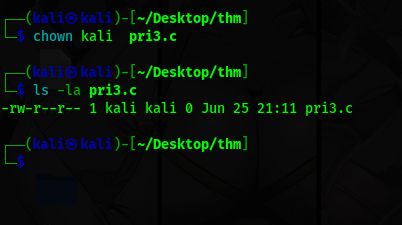
1. **Chmod:**

chmod is a common command that modifies a file or directory’s read, write, and execute permissions.



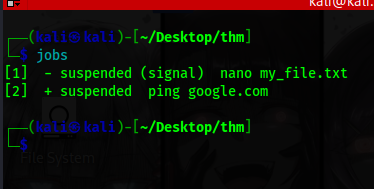
1. **Chown:**

The chown command lets you change the ownership of a file, directory, or symbolic link to a specified username.



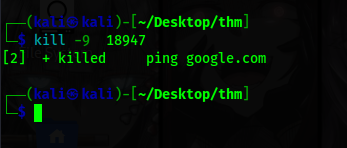
1. **Jobs:**

The jobs command will display all the running processes along with their statuses.

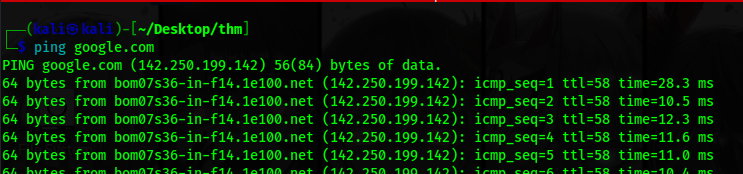


1. **Kill:**

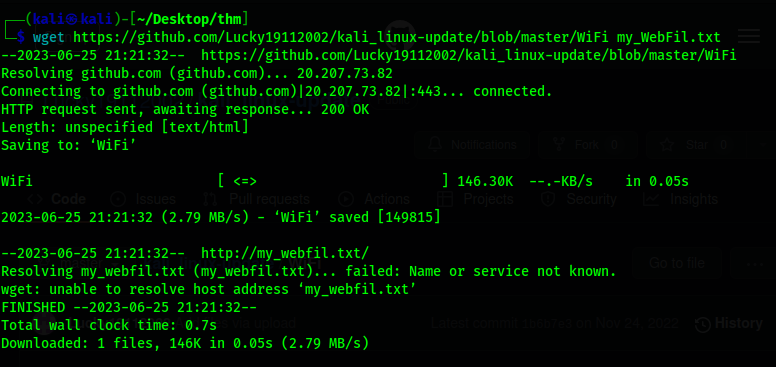
Use the kill command to terminate an unresponsive program manually.



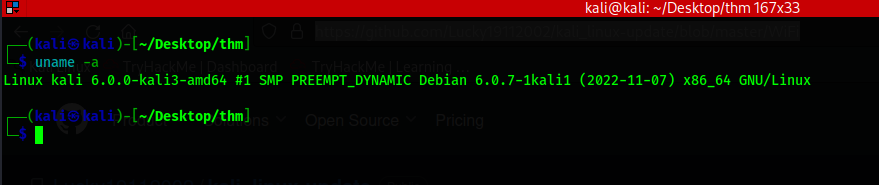
1. **Ping:**

The ping command is one of the most used basic Linux commands for checking whether a network or a server is reachable.

1. **Wget:**

The Linux command line lets you download files from the internet using the wget command.

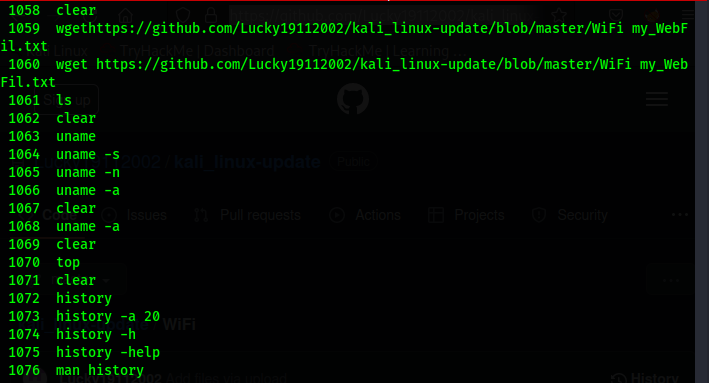
1. **Uname:**

The uname or unix name command will print detailed information about your Linux system and hardware.

1. **Top:**

The top command in Linux Terminal will display all the running processes and a dynamic real-time view of the current system.

1. **History:**

The system will list up to 500 previously executed commands, allowing you to reuse them without re-entering.

1. **Man:**

The man command provides a user manual of any commands or utilities you can run in Terminal, including the name, description, and options.

# PRACTICAL- 3

AIM: **Using Divide and Conquer Strategies design a class for Concurrent Quick Sort using C++.**

**Divide and Concure:**

The Divide and Conquer strategy is a fundamental problem-solving technique in computer science and mathematics. It involves breaking down a complex problem into smaller, more manageable subproblems, solving them independently, and then combining the solutions to obtain the final result. The strategy follows a recursive approach, where the problem is divided into subproblems, each of which is solved independently. The solutions to the subproblems are then combined to solve the original problem. This approach is particularly useful for solving problems that exhibit overlapping subproblems and can be efficiently solved using smaller instances of the same problem.

The Divide and Conquer strategy offers several advantages. It allows for efficient problem-solving by reducing the complexity of the original problem into smaller, more easily solvable subproblems. The parallelizability of the approach enables the use of concurrent or distributed computing resources, further improving the efficiency of the solution. Additionally, the modular nature of Divide and Conquer allows for code reusability and extensibility, as the same technique can be applied to various problems by appropriately defining the divide, conquer, and combine steps. Overall, the Divide and Conquer strategy is a powerful and widely applicable technique that provides elegant solutions to complex problems across different domains.

**Algorithm:**

DAC(a, i, j)

{

if(small(a, i, j))

return(Solution(a, i, j))

else

mid = divide(a, i, j) // f1(n)

b = DAC(a, i, mid) // T(n/2)

c = DAC(a, mid+1, j) // T(n/2)

d = combine(b, c) // f2(n)

return(d)

}

**Quick-Sort :**

# PRACTICAL- 5

AIM: **Write a program to check task distribution using Gprof.**

**GPROF:**

Gprof is a profiling tool for the C language that helps analyze the execution time of different functions within a program. It provides valuable insights into the performance of a program by measuring the amount of time spent in each function and identifying potential bottlenecks. Gprof works by instrumenting the program's code and gathering data on function calls and their durations. This data is then used to generate a detailed report, including a call graph that visualizes the flow of function calls and the time spent in each. By utilizing Gprof, developers can identify critical sections of their code and optimize them to improve overall program efficiency and performance.

**% time:**

This column represents the percentage of total program execution time spent in a particular function. It indicates the relative impact of each function on the overall runtime. Functions with higher percentages indicate that they consume a significant portion of the program's execution time.

**cumulative seconds:**

This column shows the cumulative amount of time spent in a function and all the functions it calls (including nested calls). It includes both the self time of the function and the time spent in its child functions. This value gives an overall measure of the total time attributed to a function and its descendants.

**self seconds:**

This column displays the exclusive time spent in a particular function, excluding the time spent in its child functions. It indicates the amount of time directly attributable to the execution of the function itself, without considering any function calls made within it.

**calls:**

This column denotes the number of times a function is called during program execution. It represents the frequency of invocation for a specific function. Functions with a high number of calls may have a significant impact on the overall performance.

**Ts/call (self):**

This column shows the average amount of time (in seconds) spent per function call for the self time. It is calculated by dividing the self seconds by the number of calls for that function. It gives an estimate of the time consumed by each individual call to the function.

**Ts/call (total):**

This column represents the average amount of time (in seconds) spent per function call, considering both the self time and the time spent in its child functions. It is calculated by dividing the cumulative seconds by the number of calls for that function. This value provides an estimate of the time spent per call when taking into account the function's overall impact.

**name:**

This column displays the name of the function being analyzed. It helps identify which specific function is associated with the corresponding data in the other columns.

**number of samples:**

This column, when available, provides the number of times a function was sampled during program execution. It indicates the reliability of the profile data. Functions with a higher number of samples have more accurate timing information and can be considered statistically significant.

**self %:**

This column represents the percentage of total program execution time that is attributed to the self time of a function. It helps identify functions that have a significant impact on the overall runtime in terms of their own execution time. Functions with a high self percentage may be good candidates for optimization.

**children seconds:**

This column shows the total time spent in the child functions called by a particular function. It indicates the amount of time spent within the function's own scope, excluding any time spent in its descendants. It can help identify functions that have a significant impact on the overall runtime due to the time spent in their child functions.

**children %:**

This column displays the percentage of total program execution time that is attributed to the time spent in the child functions called by a specific function. It indicates the impact of the child functions on the overall runtime of the parent function. Functions with a high children percentage may indicate potential optimization opportunities within the child functions..

**index:**

This column represents the index number assigned to each function in the profile. It can be used to sort and organize the functions based on their index values. This index can be helpful when comparing different profiles or when tracking changes in the codebase over time.

**Msecs:**

In some Gprof profiles, this column represents the time spent in a function in milliseconds. It provides an alternative representation of the time measurements, particularly if the time values are small and need more precision.

**source file:line:**

When available, this column displays the source file and line number where the function is defined. It helps locate the function in the source code for further analysis and optimization. It allows developers to directly navigate to the specific code that needs attention.

**Simple Code GPROF:**

**Program:**

#include <stdio.h>

int addNumbers(int a, int b) {

return a + b;

}

int main() {

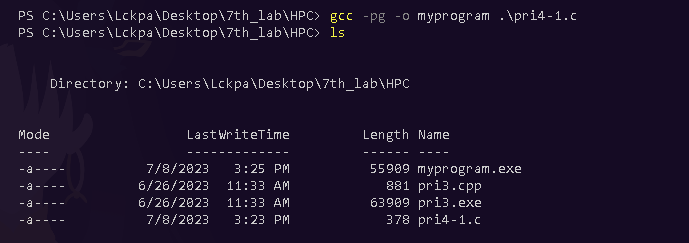
int num1 = 5;

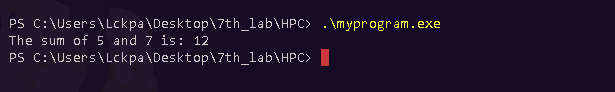
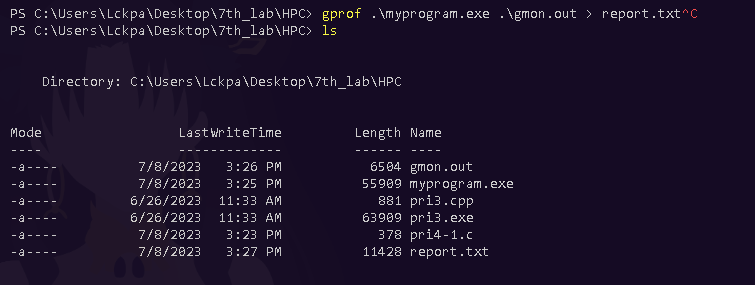
int num2 = 7;

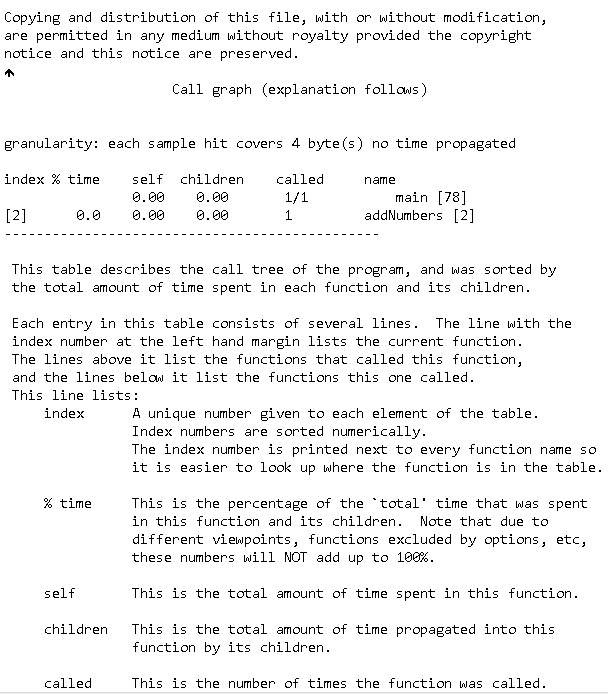
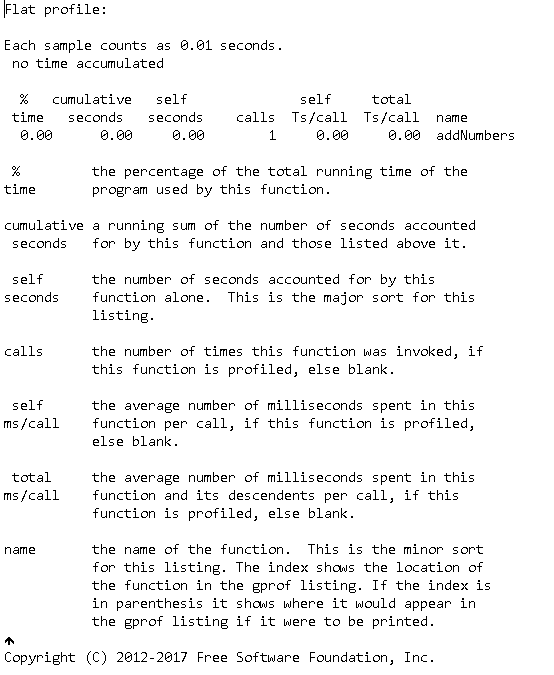
int sum = addNumbers(num1, num2);

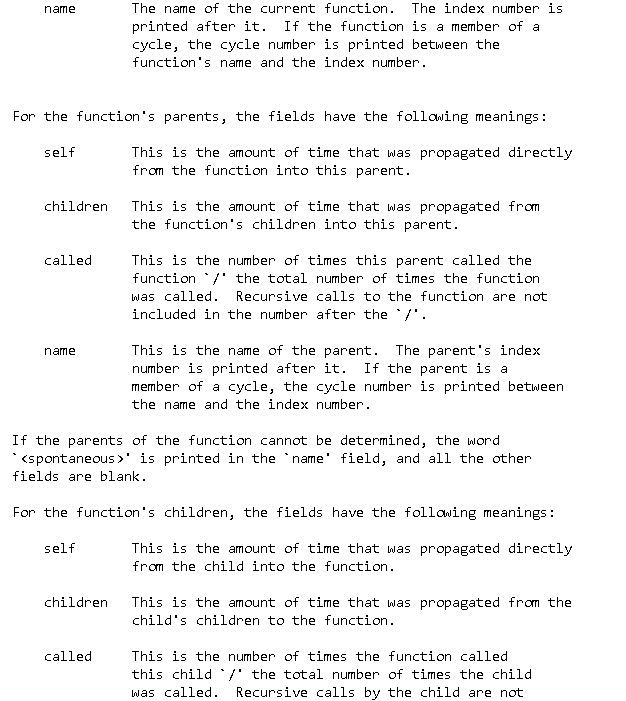
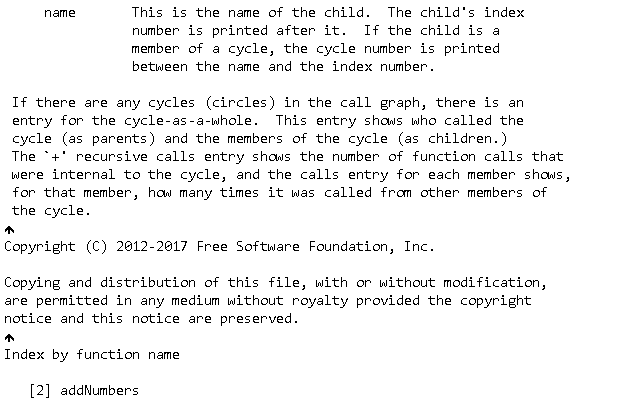
printf("The sum of %d and %d is: %d\n", num1, num2, sum);

return 0;

****}



******Output:**



**Complex Code GPROF:**

**Program:**

#include <stdio.h>

#include <time.h>

void timeConsumingTask() {

for (int i = 0; i < 1000000000; i++) {

// Perform some computations

} }

int main() {

clock\_t start = clock();

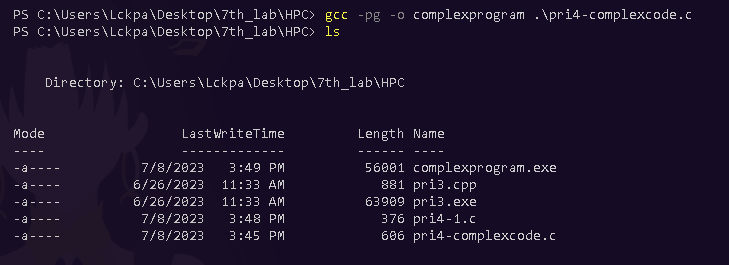
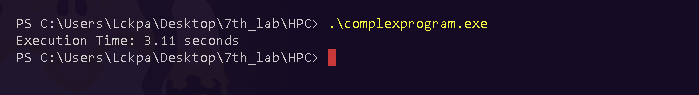
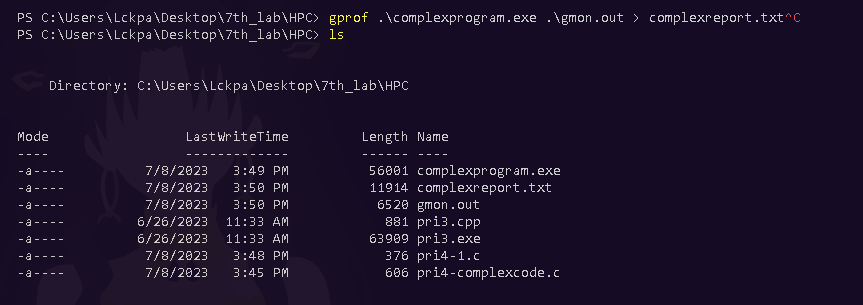
timeConsumingTask(); // Perform time-consuming task

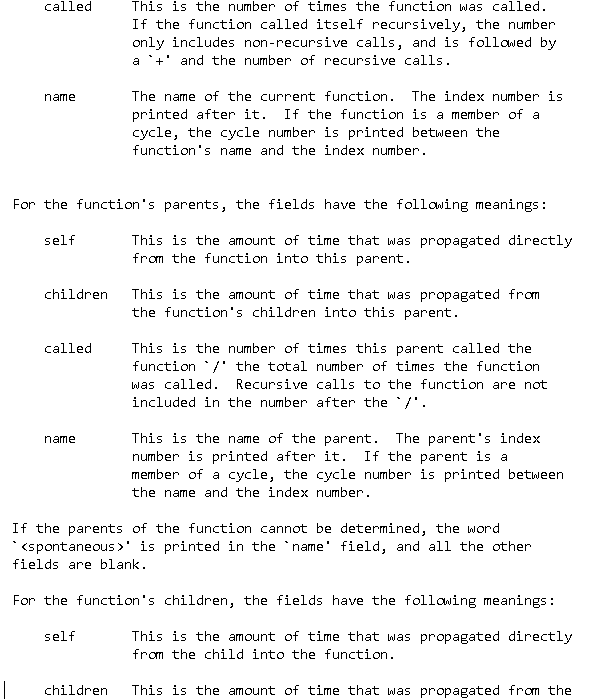
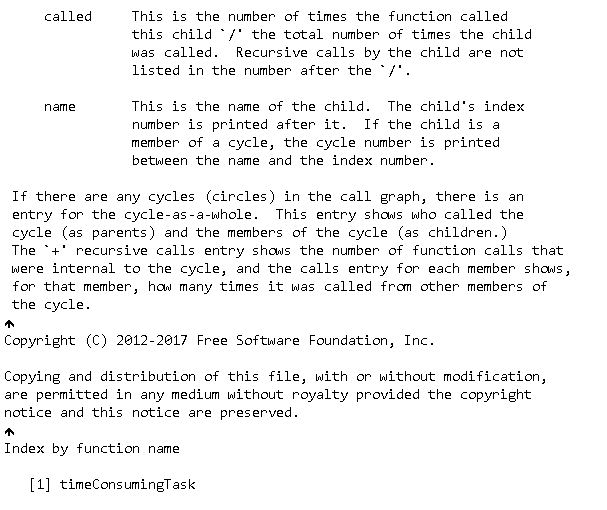
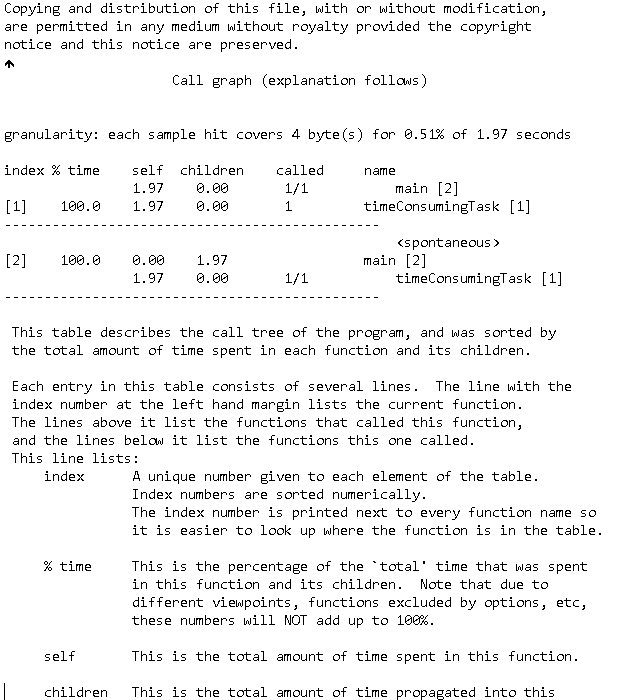
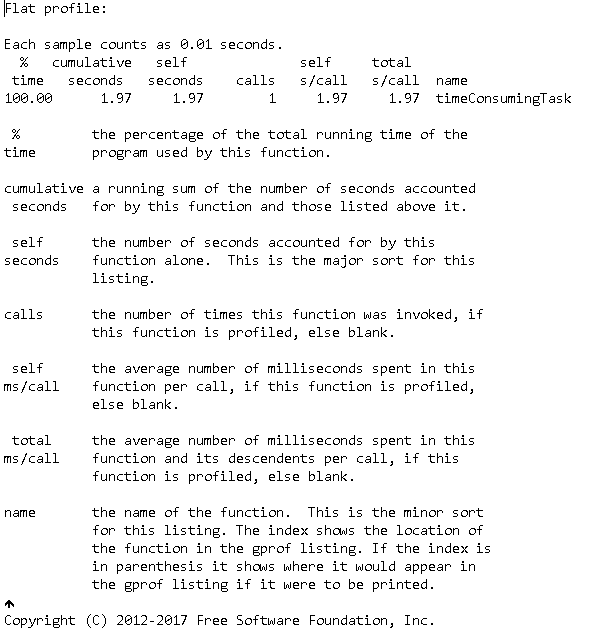
clock\_t end = clock();

double executionTime = (double)(end - start) / CLOCKS\_PER\_SEC;

printf("Execution Time: %.2f seconds\n", executionTime);

return 0;

}

******Output:**